Born to Trade: a Genetically Evolved Keyword Bidder for Sponsored Search

Michael Munsey, Jonathan Veilleux, Sindhura Bikkani, Ankur Teredesai, and Martine De Cock

***Abstract*— In sponsored search auctions, advertisers choose a set of keywords based on products they wish to market. They bid for advertising slots that will be displayed on the search results page when a user submits a query containing the keywords that the advertiser selected. Deciding how much to bid is a real challenge: if the bid is too low with respect to the bids of other advertisers, the ad might not get displayed in a favorable position; a bid that is too high on the other hand might not be profitable either, since the attracted number of conversions might not be enough to compensate for the high cost per click.**

In this paper we propose a genetically evolved keyword bidding strategy that decides how much to bid for each query based on historical data such as the position obtained on the previous day. In light of the fact that our approach does not implement any particular expert knowledge on keyword auctions, it did remarkably well in the Trading Agent Competition at IJCAI2009.

I. Introduction

The use of sponsored search auctions is one of the key mechanisms by which search engines generate profit. Advertisers can buy space on the web pages produced by popular search engines and place advertisements to promote their products alongside the regular algorithmic search results. The allocation of these advertising slots and their pricing is done via auctions. Since the introduction of this concept in 1998, sponsored search has evolved into a major source of revenue for internet giants such as Google, Yahoo!, Bing and others. Its success can be attributed partly to its effectiveness as a form of highly targeted advertising, and partly to the appealing framework that allows even small-scale advertisers to use it easily and effectively while only paying when their ad is clicked upon. When a user enters a keyword query into a search engine, he gets back a page with results, containing both the links most relevant to the query and the sponsored links, i.e., paid advertisements. When a user clicks on a sponsored link, he is sent to the respective advertiser's web page. The advertiser then pays the search engine for sending the user to his web page.

As Figure 1 illustrates, sponsored links are typically clearly distinguishable from the actual search results. However, the visibility of a sponsored search link depends on its location (slot) on the result page. Typically, a number of advertisers naturally prefer a slot with higher visibility, i.e., towards the top of the list of sponsored links. Hence, search engines need a system for allocating the slots to advertisers and deciding on a price to be charged to each advertiser. Due to increasing demands for advertising space, most search engines are currently using auction mechanisms for this purpose. These auctions are called sponsored search auctions. In a typical sponsored search auction, advertisers are invited to submit bids on keywords, i.e., the maximum amount they are willing to pay for a user clicking on the advertisement displayed on the search results page when a user submits a query containing those keywords. Based on the bids submitted by the advertisers, the search engine picks a subset of advertisements along with the order in which to display them. The actual price charged, i.e., the cost per click (CPC), also depends on the bids submitted by the advertisers.

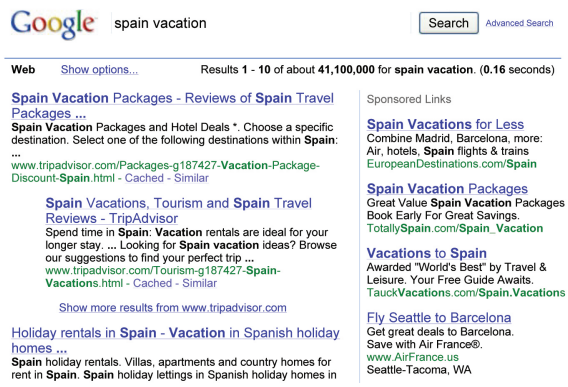

Fig. 1. Result of a search performed on Google for the query *spain vacation*. The sponsored search results (on the right) are clearly distinguishable from the algorithmic search results (on the left).

Due to the limited budget of each advertiser, strategic bidding behavior plays a crucial role in sponsored search auctions. From an advertiser's perspective, an ideal strategy has to prevent the advertiser from overbidding, while at the same time bidding enough to obtain his favorite position, thereby beating competing advertisers. To encourage the research community to develop such bidding strategies, in 2009 a new game was introduced in the Trading Agent Competition (TAC), a series of international research tournaments to promote research and education in the technology underlying trading agents [8]. Agents in the TAC Ad Auctions game[1] (TAC/AA), representing internet advertisers, bid for search engine advertisement placement over a range of interrelated keyword combinations. A back-end search user model trans-

All authors are with the Institute of Technology, University of Washington, 1900 Commerce Street, Tacoma, WA-98402, USA (email: {munseym, veillj, sindhub, ankurt, mdecock}@u.washington.edu).

Martine De Cock is on leave from the Department of Applied Mathematics and Computer Science, Ghent University, Krijgslaan 281 (S9), 9000 Gent, Belgium (email: martine.decock@ugent.be).

[1] http://aa.tradingagents.org/

lates placement over each simulated day to impressions, clicks, and sale conversions (purchases), yielding revenue for the advertiser. Advertiser strategies need to combine online data analysis and bidding tactics to compete for maximum profit over the simulated campaign horizon.

In this paper we describe an approach with which we participated in the final tournament of the game, held at the workshop on Trading Agent Design and Analysis (TADA-09) at IJCAI2009. In our approach, we model an agent as a finite state machine. At any given time the agent is in a particular state, defined by a set of parameters that we believe are most crucial in making a decision about a bid. When asked for a new bid for a particular query, the agent decides to either stick to, or to adapt the previous bid for that query, and, if it deems a change of the bid is needed, by how much. The decisions that an agent makes are controlled by the state that the agent is in; this state is in turn determined by the agent's past behavior and the reaction of the market to that behavior. In that sense, the bid that an agent makes indirectly transitions it into another, typically different, state of the finite state machine. We use an evolutionary approach to develop finite state machines (agents) that perform well. The performance of agents in intermediate generations is evaluated by letting them compete in game instances which are run specifically for the fitness computation. Agents that accumulate a higher average profit are more likely to contribute offspring to the next generation.

The remainder of this paper is structured as follows: in Section II, we provide more details about sponsored search and the ad auctions game. In Section III we describe our approach, while in Section IV we present empirical results, including results about the performance of our agent at the final tournament of the TAC Ad Auctions game held at the TADA-09 workshop at IJCAI2009. Finally, in Sections V and VI, we respectively present related work and ideas for future research.

II. Problem Description

A. Sponsored Search

Advertisers spend billions of dollars every year on search engine marketing (SEM), a form of internet marketing that seeks to promote web sites by increasing their visibility in search engine result pages. In contrast to search engine optimization (SEO), which is the process of improving the volume or quality of traffic to a web site from search engines via pure search results (also known as organic or algorithmic search results), SEM deals with paid inclusion. The largest SEM vendors are Google AdWords[2], Yahoo! Search Marketing[3], and Microsoft adCenter[4], all of which provide cost per click (CPC) advertisements through sponsored search auctions.

There are three entities involved in a sponsored search scenario: a user who generates queries, clicks ads, and purchases products; a publisher (search engine) who receives queries from users, matches queries to ads, and holds position auctions for display; and an advertiser who bids for position ranking and selects ads for display. The relationship between these entities can be beneficial to all involved. Because of the cost of a click, an advertiser does not want a user to see or click an advertisement that is not likely to lead to a conversion, likewise a user does not want to be bothered by advertisements that they are not interested in. Both the advertisers and the users must have a level of trust with the publisher, or they may leave the system in favor of an alternate publisher.

Given the bids of all advertisers for a particular query, it is up to the publisher to decide which slot to allocate to which advertiser. Ranking advertisements in response to a query is a complicated task that involves ideas from information retrieval as well as insights from microeconomics. Indeed, it is in the interest of a search engine to give more visibility to advertisements from higher bidders because this will lead to a higher short term CPC revenue for the search engine. At the same time however, the top ranked ads should be sufficiently relevant to the query to please users who have come to expect highly relevant search results; irrelevant sponsored search results might disappoint users who will then leave the system in favor of another, thereby jeapordizing the long term revenue of the search engine.

Another decision to be made by the publisher is how much to charge advertisers for a click on an advertisement in a particular slot. The most straightforward way is to set the CPC for a slot at the price bid by the winner of that slot (called the generalized first price rule). The major search engines are however known to employ a generalized second price rule, a generalized variant of a Vickrey auction in which the CPC for a slot is determined by the price bid of the winner of the next-best position [4].

In this paper however, we look at the sponsored search problem from the perspective of an advertiser. For any mechanism followed by the publisher, advertisers face the problem of how to generate bids over time in a competitive and highly dynamic ad market. We assume that the advertiser has already selected a set of keyword queries that he wishes to place bids on. Our contribution and focus is on solving the problem of coming up with the best bids for these selected keywords. Specifically, we focus on the genetic encoding of the profit maximization problem to ensure optimal return on investment accounting for historical bid patterns. Note that this problem is different from identifying the most profitable set of keywords among millions of available keywords as is done e.g. in [12]. In an actual search engine marketing campaign, both problems need to be solved.

B. The Ad Auctions Game

The TAC/AA scenario is designed to include many of the interesting strategic aspects of sponsored search auctions, while being repeatable and computationally amenable to empirical analysis. Recall that there are three entities involved in a sponsored search scenario: users, advertisers, and a

[2]http://adwords.google.com
[3]http://searchmarketing.yahoo.com
[4]https://adcenter.microsoft.com/

—, —	(F0)	Lioneer, Audio	(F2)
Lioneer, —	(F1)	Lioneer, DVD	(F2)
PG, —	(F1)	PG, TV	(F2)
Flat, —	(F1)	PG, Audio	(F2)
—, TV	(F1)	PG, DVD	(F2)
—, Audio	(F1)	Flat, TV	(F2)
—, DVD	(F1)	Flat, Audio	(F2)
Lioneer, TV	(F2)	Flat, DVD	(F2)

Fig. 2. There are 16 distinct query kinds in the TAC/AA game environment.

publisher (or search engine) that brings them together. The game environment simulates the behavior of the users and the search engine: the search users and the publisher follow fixed (stochastic) policies built into the game environment. The advertiser agents, on the other hand, follow policies implemented by competition entrants.

In the game, participating advertisers have an interest in the keywords *Lioneer, PG, Flat, TV, Audio,* and *DVD*. The first three correspond to the names of manufacturers, while the others are components in the home entertainment market. The game environment simulates users who launch queries that contain one, two, or none of these keywords. Mentioning neither a component nor manufacturer is called an F0 level query. Mentioning one or the other, but not both, is denoted an F1 level query. Mentioning both component and manufacturer is called an F2 level query. In total, there are 16 distinct kinds of queries, as illustrated in Figure 2.

A game instance corresponds to a period of 60 days[5]. At the beginning of each day, each participating advertiser submits a bid for each of the 16 query kinds from Figure 2. In addition, each advertiser also informs the publisher of the kind of ad to display for each of the 16 query kinds: either a targeted ad mentioning a particular product, or a generic ad. Finally, advertisers may also submit individual daily spend limits for each of the 16 query kinds, as well as a limit for the aggregate spend across queries. When a spend limit is reached, the affected ads will no longer be shown to users for the rest of the current day.

After all advertisers have submitted their bids for the day, the publisher executes an ad auction for each query kind to determine the ad rankings and click prices. Users then issue queries, receive results, and consider clicking on ads. When a user clicks on an ad, the user is taken to the respective advertiser's landing page. In addition, the advertiser is charged a cost per click that was determined by the publisher when the ad auction was run. When a user visits an advertiser's landing page, he determines whether or not to purchase a product from that advertiser. If the user purchases, the event is termed a conversion, and the advertiser earns a profit. As new queries are launched throughout the day, the publisher monitors the spend limits of the advertisers and reruns ad auctions as necessary.

For a more detailed description of the stochastic policies governing the behavior of the publisher and the users, we refer to the game specifications [7]. An interesting fact to point out is that every simulated user in the game has a particular product preference (say *Lioneer, TV*). A user is more likely to click on a targeted ad that matches its preference than on a generic ad, but a user is also more likely to click on a generic ad than on a targeted ad that does *not* match its preference. Hence an advertiser needs to decide with care whether it wants a generic or a targeted ad to be displayed for a given query kind.

Although every advertiser sells every product, there are two differences between individual advertisers, assigned by the game environment at the beginning of each game instance. First, each advertiser specializes in a particular manufacturer and a particular component (say *PG, TV*). The specialization affects conversion probability and sales profit. Users with a component preference matching the advertiser's specialization (e.g. *TV*) are more likely to purchase once they reach the advertiser's landing page, i.e., the odds of converting are increased. Furthermore, an advertiser receives a manufacturer specialist bonus for every sale of a product from the manufacturer matching its specialization (e.g. *PG*). Second, each advertiser is assigned a distribution capacity of low, medium, or high. This is used to model the phenomenon that if advertisers sell too much products in a short period, their inventories run short and they have to put items on backorder. As a result, shoppers will be less inclined to purchase, and conversions suffer, i.e. the odds for converting decrease. Roughly speaking, the distribution capacity of an advertiser directly relates to the number of products he can sell in a sliding aggregation window of $W = 5$ days without risking a backorder penalty. For more details, we refer to [7].

Finally, at the beginning of each day, each advertiser receives three reports based on events from the previous day, namely a daily query report from the publisher, a daily account status report from the bank, and a daily sales report from the sales analyst enumerating the sales for each of the 16 query kinds. For the full details of the contents of these reports, we refer again to the game specifications [7]. Important for the approach we describe in the next section, is that the daily query report for an advertiser contains, among other information, the average position of the slot obtained by that advertiser for each of the 16 query kinds. The reports about day $d-1$ are only made available *after* advertisers have placed their bids for day d. As a result, information about events that took place on day $d-1$ can be used at the earliest to influence bids for day $d+1$.

III. Evolving a Bidding Strategy

Recall that at the start of day d, an advertiser needs to submit a bid bundle specifying its CPC offer and ad choice for each of the 16 query kinds, to be applied on that day. Furthermore, advertisers may also submit individual daily spend limits for each of the 16 query kinds, as well as a limit for the aggregate spend across queries. This section is almost entirely devoted to how to decide the CPC offer, which is the heart of our approach. In Section III-A, we

[5]In the game environment, the length of such a "day" takes only a few seconds in real time.

explain how the bidding behavior of an advertiser can be encoded as a real valued chromosome, while in Section III-B we describe the details of our genetic algorithm, including fitness computation and reproduction. Finally, in Section III-C we briefly address the issues of ad choice and spend limits, which are not part of our genetic algorithm.

A. Encoding

In our approach, the bidding behavior of an advertiser is governed by a finite state machine with 7 different states. The state that an advertiser x is in at the beginning of day d depends on its backorder on day $d-1$, which we compute as

$$B_{d-1}(x) = \sum_{i=d-W}^{d-2} c_i(x) - C(x) \tag{1}$$

with $C(x)$ the distribution capacity of the advertiser and W the length of the aggregation window. In the game environment, at the beginning of a game instance, an advertiser is assigned a distribution capacity of either $C(x) = 300$ units (low), 400 units (medium), or 500 units (high), corresponding to the number of units he can sell in a sliding window of $W = 5$ days without the risk of a backorder penalty. The more an advertiser exceeds his distribution capacity within an aggregation window, the smaller the odds that users will place additional orders. In (1), $c_i(x)$ denotes the total number of conversions of advertiser x on day i. This information is readily available from the daily report of the sales analyst. For values of i smaller than 1, we set $c_i(x) = 0$, indicating that no units have been sold before day 1, which is the start of a game instance. Note that the summation in (1) does not include the number $c_{d-1}(x)$ of conversions on day $d-1$. As explained above, the report for day $d-1$ becomes available only after the bids for day d have been placed. In that sense, the formula on the right hand side of (1) only provides a lower bound for the true backorder at the end of day $d-1$.

The backorder computed according to (1) can be positive or negative. A positive value indicates that more conversions have been made than the distribution capacity allows, resulting in a true backorder. A negative value means that there is room for more conversions before the distribution capacity over the aggregation window is reached. To facilitate encoding, we discretize the range for backorder into 7 intervals over the integers, each corresponding to a state in the finite state machine. The state $S_d(x)$ of advertiser x at the beginning of day d is defined as:

$$S_d(x) = \begin{cases} s_1, & \text{if } B_{d-1}(x) \leq -101 \\ s_2, & \text{if } -100 \leq B_{d-1}(x) \leq -51 \\ s_3, & \text{if } -50 \leq B_{d-1}(x) \leq -26 \\ s_4, & \text{if } -25 \leq B_{d-1}(x) \leq -11 \\ s_5, & \text{if } -10 \leq B_{d-1}(x) \leq 0 \\ s_6, & \text{if } 1 \leq B_{d-1}(x) \leq 10 \\ s_7, & \text{if } 11 \leq B_{d-1}(x) \end{cases} \tag{2}$$

Next, every state contains an action table that tells the advertiser which bid $b_x(q,d)$ to place for each query q on day d. This decision depends on the previous bid of the advertiser for that query q, the average slot position previously obtained for query q, as well as how well query q matches the advertiser's specialization.

The average slot position obtained for a query on a given day is stated in the daily report from the publisher as a double in the range $[1, 8]$. For instance, if an advertiser has a position value of 2 for query —, *Audio*, this means that the advertiser's ad, on average, was in the second position of the search results page for query —, *Audio* on that day. It is an average because the position of the advertiser's ad may vary during the course of a day for each query, as advertisers might opt out because their spending limit is reached. In the game, a better position generates more impressions. For our purposes, we distinguish between 6 possible values for position: $1, 2, 3, 4, 5$, and > 5. Value > 5 means that the ad did not get a slot position in the top 5, which might mean that it got a lower ranked slot, or no slot at all. We do not distinguish further between the cases comprised by > 5 in order to keep the size of the action table manageable.

Specialty matching represents the degree to which a query matches the advertiser's specialization. A keyword from the query that occurs in the advertiser's specialization is called a hit; e.g. *Lioneer* is a hit in *Lioneer, Audio*. A keyword from the query that does not occur in the advertiser's specialization is called a miss. The level of matching between a query and the advertiser's specialization is defined as follows:

2 misses:	VERY BAD
1 miss, 0 hits:	BAD
1 miss, 1 hit:	WEAK
0 hits, 0 misses:	NEUTRAL
1 hit, 0 misses:	GOOD
2 hits:	PERFECT

For example, if our advertiser's specialization is *Lioneer, Audio*, then query *PG,* — is considered a BAD match, while *Lioneer, TV* and *PG, Audio* are WEAK matches. The query —, *Audio* on the other hand is considered to be a GOOD match. The queries —, — and *Lioneer, Audio* are respectively the only NEUTRAL match and PERFECT match. *PG, TV* would be an example of a VERY BAD match.

An action table is a two dimensional array in which the rows correspond to slot positions, and the columns correspond to speciality matchings. As an example, Table I depicts the action table contained in state s_6 of the agent with which we participated in the 2009 TAC/AA competition. The entries in the table are numbers in the interval $]-1, 1]$ that dictate how the previous bid $b_x(q, d-1)$ for query q should be adjusted to obtain the new bid $b_x(q,d)$. In particular,

$$b_x(q,d) = (1 + A^s_{i,j}) \cdot b_x(q, d-1) \tag{3}$$

where $A^s_{i,j}$ is the number on row i and column j of the action table of advertiser's x current state s. The relevant i is determined based on the average slot position previously obtained for query q, while j corresponds to the match between query q on one hand, and the specialization of advertiser x on the other hand.

TABLE I
ACTION TABLE FOR BACKORDER $B_{d-1}(x) \in [1, 10]$.

	VERY BAD	BAD	WEAK	NEUTRAL	GOOD	PERFECT
1	−0.2411	−0.3492	0.6218	0.3053	0.2174	0.0054
2	−0.0767	0.0969	−0.2324	−0.2929	0.1425	0.0722
3	0.0233	0.2947	0.1039	0.0412	−0.2582	0.0524
4	0.0549	−0.2146	−0.1124	0.1133	0.0028	0.4546
5	−0.1486	0.2938	−0.0737	−0.0257	−0.4715	−0.0873
> 5	0.1214	−0.1931	0.2013	0.1824	−0.0449	−0.0769

The strategy described above relies on knowing in which state the advertiser is at any given moment, which is determined with (1). Note that the first time that (1) uses any information about the market, other than the advertiser's own distribution capacity $C(x)$, is for $d = 3$. Recall that the daily reports for day $d = 1$ only become available *after* the advertisers have placed their bids for day $d = 2$, so day $d = 3$ is the earliest point in the game at which advertisers can make some sort of informed decision. In other words, CPC offers for days $d = 1$ and $d = 2$ can be considered shots in the dark. In Section IV we mention the CPC offers that our agent in the 2009 TAC/AA competition used for the first two days, applying the heuristic of bidding more for queries that are good matches w.r.t. the advertiser's specialization, than for queries that are bad matches. After day 2, adjustments are made to these bids through the procedure outlined above.

B. Training

As became clear above, the behavior of an advertiser in our approach is governed by a finite state machine with 7 states, each of which contains a 6 by 6 action table. The 252 values (= $7 \cdot 6 \cdot 6$) needed to fill the action tables are learned through an evolutionary process, in which an advertiser is encoded as a real valued chromosome of length 252.

a) Fitness function: The fitness of a chromosome is defined as the average profit earned by the chromosome in the game. Hence determining the fitness of a chromosome requires running one or more game instances, which makes evaluating the fitness a time consuming process. In practice, we compute the fitness of a chromosome based on at least two rounds of the game.

b) Initial population: To start, we create a population of size 16. Since in the standard game setting the number of advertisers per game instance is 8, a population size of 16 allows us to have every chromosome compete exactly twice in a total of 4 game instances (in a training scenario where we evolve our advertiser agents by making them compete against each other, and no external agents are involved). All genes in 15 of the chromosomes are initialized with uniformly random values in the range $]-1, 1]$. The remaining chromosome is initialized as a constant bidder, i.e., all of its genes are set to 0.0. The motivation behind seeding the initial population with a constant bidder is to allow the population to learn from and become at least as good as an agent that bids the same amount for the duration of a game.

c) Selection: The fittest chromosome is carried over automatically to the next generation. Furthermore, we use roulette wheel selection to pick 15 pairs of chromosomes for reproduction. In our implementation, a chromosome may be picked more than once and may even be in a pair with itself.

d) Reproduction: Each pair of chromosomes produces a single child. Because our alleles are real values, we use BLX-α crossover, an extension of flat crossover [6] in which each child allele is chosen as a uniformly distributed random value from the interval

$$[\min(x_1, x_2) - I \cdot \alpha, \max(x_1, x_2) + I \cdot \alpha] \qquad (4)$$

where x_1 is the corresponding first parent's allele, x_2 is the corresponding second parent's allele, and $I = \max(x_1, x_2) - \min(x_1, x_2)$. The value of α determines how far outside the interval between the parents' alleles the child's allele can be. In our case, $\alpha = 0.25$ was chosen so that the majority of new alleles are within the parent's interval, while still maintaining diversity in the population.

e) Mutation: Next, we mutate each gene in the newly created pool of children with probability 0.02. When a gene is mutated, its allele is replaced with a new random value in the interval $]-1, 1]$. Note that we carefully avoid -1 as a bid adjustment value both when setting up the initial population and when mutating. The reason is that $b_x(q, d)$ becomes 0 when $A^s_{i,j} = -1$ in (3). This means that advertiser agent x would bid 0 for query q, not only on day d, but permanently from then on, because there is no way to alter a zero bid into a different bid by multiplication. Therefore, we do not introduce bid adjustments of -1 in the initial population nor through mutation; the crossover process will also not introduce the value of -1.

f) Generations and termination: The fittest chromosome and the 15 children become the new population. The algorithm then proceeds to the next iteration, or generation, starting with evaluation of fitness. The algorithm runs for a specified number of generations. The time to process one generation is most affected by the time to run a tournament, which on an Intel Quad Core Q9550 2.83 Ghz CPU with 8 Gb DDR2 RAM is approximately 4 minutes for a tournament consisting of 4 games. For 200 generations, this is more than 13 hours.

C. Ad Choice and Spend Limits

While considering advertisements sequentially, the probability that a user will click on a particular advertisement is modified by a *targeting factor* in the game environment. The targeting factor f_{target} gives a bonus to a targeted advertisement that matches the user's preference, and a penalty if it does not (TE is given as 0.5, see [7]):

$$f_{target} = \begin{cases} 1+TE, & \text{if targeted ad, matches} \\ 1 & \text{if generic ad} \\ \dfrac{1}{1+TE} & \text{if targeted ad, does not match.} \end{cases}$$

There are 9 possible user preferences, one for each product-manufacturer combination. A user only queries for his preference, e.g. a user with preference *Lioneer, DVD* only launches the queries *Lioneer, DVD*; *Lioneer,*—; —, *DVD*; and —, —. For an F2 query, we know precisely the user's preference, and use a targeted advertisement that matches it. For an F0 query, we know nothing about the user's preference. Random guessing would be correct 1/9 of the time, which would incur more penalty than bonus. For an F1 query random guessing would be correct 1/3 of the time, but with both bonuses and penalties applied, on average would only yield 17/18 of the targeting factor that we would get with a generic advertisement. Without further modeling and prediction of users' preferences, it is not profitable to use targeted advertisements for either F0 or F1 queries, therefore our advertiser agents use the generic advertisement in both cases.

Finally, an advertiser in our approach does not set any daily spend limits, i.e. we do not specify a limit to the maximum CPC that can be accumulated for an ad, after which the affected ad will no longer be shown to users for the current day. Instead, our advertiser agent achieves the effect of spending less money on advertisements or getting less conversions (which might be desirable in case of a high backorder) by decreasing its bids.

Because the finite state machine described in Section III-A does not decide on an exact bid, but rather on a bid change which can be at most 100% different from the previous bid, we decided to set some common sense limits to bid amounts. It is easy to reach a small bid in only one day with a large negative bid adjustment, but it could take several days to return to the original value because a bid can at most double in one day (e.g. a bid of 5 can be decreased by 90% to arrive at a bid of 0.5 in one day, after which it will take at least 3 days to bring the bid back up to 4). This makes extremely low bids undesirable because it is very hard to recover from them in a timely manner. Extremely high bids can also be detrimental because the delay between bid submission and reports on the results of the bid can cause the advertiser agent to not reduce the bid as quickly as it should, potentially loosing a lot of money in the process. For these reasons, we set a lower and upper limit on the bid amounts. The limits used by our agent in the 2009 TAC/AA competition are given in Section IV.

TABLE II

NUMBER OF TIMES, AS A PERCENTAGE, THAT AGENTS FROM AN EVOLVED POPULATION OBTAINED A PARTICULAR SLOT POSITION FOR A PARTICULAR SPECIALIZATION MATCHING.

	1	2	3	4	5	>5
PERFECT	4.33	2.12	0.82	0.30	0.17	0.23
GOOD	2.09	2.05	2.14	1.87	1.47	2.03
NEUTRAL	0.08	0.14	0.17	0.21	0.38	2.98
WEAK	1.75	1.68	1.81	2.05	2.78	12.78
BAD	2.30	3.44	3.75	3.76	4.17	11.22
VERY BAD	1.53	2.17	2.75	3.49	3.71	11.26

IV. RESULTS

As described in Section III-B, we evolve a population of 16 advertiser agents by having them compete against each other. We compute the fitness of an agent as its average profit over two game instances, requiring 4 game instances per generation.

Table II shows, as a percentage, how many times agents from an evolved population obtained a particular slot position for a particular specialization matching. The data is generated from games played by the agents in generation 15 through 20. Table II clearly shows that over time agents learn to prefer and obtain higher positions for queries that provide bonuses, and lower positions for queries that do not. For instance, as the first row indicates, in 7.97% (= 4.33+2.12+0.82+0.30+0.17+0.23) of the situations considered, the agent needed to place a CPC offer to obtain a slot position for a query that is a PERFECT match with its specialization. In more than half of these occasions, the agent obtained first position, and in more than 4 out of 5 of these occasions, the agent obtained either first or second position. For WEAK, BAD, and VERY BAD specialization matches, we observe the opposite phenomenon, i.e., the agent is most likely to not be in the top five positions.

Note that the agents learned to behave like this through the evolutionary process, and that they have not been told in advance for which specialization matchings they should try to obtain top ranked slot positions. The labels we gave to the different specialization matchings in Section III-A might be misleading in this sense as they seem to indicate a preference of the different specialization matchings, suggesting that e.g. a NEUTRAL match is potentially more profitable than a BAD match. While it is indeed very reasonable to assume that a PERFECT match can lead to more profit than a VERY BAD match, and hence we should expect an agent to really try to achieve a top ranked position for a PERFECT match while making less efforts for a VERY BAD match, the picture is not so clear in advance for some of the other specialization matchings. It is interesting to observe for instance that the agents learned that queries corresponding NEUTRAL matchings are not very profitable.

In early runs of our genetic algorithm, the initial bid of each agent for each query was set to .75. With the initial bids all equal, we discovered that by the end of games agents

TABLE III

OUR APPROACH (DNAGENT) OBTAINED 3TH PLACE AMONG THE 15 TEAMS PARTICIPATING IN THE SEMIFINALS OF THE 2009 TAC/AA COMPETITION.

Semifinal Results		
Place	Agent	Average Profit
1	TacTex	70,690
2	Schlemazl	68,928
3	**DNAgent**	**68,273**
4	epflagent	68,224
5	AstonTAC	66,452
6	UMTac09	65,574
7	QuakTAC	64,136
8	MetroClick	63,043
9	Mertacor	62,533
10	UWTAgent	62,323
11	Merlion	61,274
12	Bishop	59,310
13	McCon	41,728
14	CrocodileAgent	34,554
15	WayneAd	19,185

TABLE IV

OUR APPROACH (DNAGENT) OBTAINED 5TH PLACE AMONG THE 8 TEAMS THAT ADVANCED TO THE FINALS OF THE 2009 TAC/AA COMPETITION.

Final Results		
Place	Agent	Average Profit
1	TacTex	79,886
2	AstonTAC	76,281
3	Schlemazl	75,408
4	QuakTAC	74,462
5	**DNAgent**	**71,777**
6	epflagent	71,693
7	MetroClick	70,632
8	UMTac09	66,933

learned to bid more for queries matching their specialization and less for queries corresponding to poor matches (which is in line with the observations described above). Based on this information, we made a one-time adjustment, setting the initial bids for queries in each specialty matching group to match the average ending bid for those queries over several games. In particular, we used the following initial bid values for our agent in the 2009 TAC/AA competition: 0.33 for VERY BAD, 0.50 for BAD, 0.75 for WEAK, 0.50 for NEUTRAL, 1.25 for GOOD, and 1.50 for PERFECT.

For reasons explained in Section III-C, we imposed a lower and upper limit on the bids placed by our agent. We set the lower limit to 0.02, which is the lowest possible value of the regular slot reserve score according to the game specifications, meaning that bids under 0.02 are never taken into account anyway in the ad ranking process. We allowed bids up to 2.54 for PERFECT matches, 0.8 for NEUTRAL matches (since the typically low number of conversions makes it hard to make money out of these) and 1.5 for all the rest. We found that in more than 90% of the cases our trained agents spontaneously bid in between these limits.

To select an agent for participation in the 2009 TAC/AA competition, we ran our genetic algorithm until it reached termination based on the shrinking diversity of the population. We repeated this process 8 times, thereby producing 8 candidate agents, i.e. the winners of each of those 8 runs of the genetic algorithm. Finally we had these 8 candidate agents compete in a single tournament against each other to determine which one to enter in the 2009 TAC/AA competition.

Tables III and IV show the performance of our agent[6] at the 2009 TAC/AA competition. Notice how our profit increased between the semifinals and the finals, while our position decreased. One explanation could be that the agents who did not advance to the finals may have tended to bid too high, and it is possible that our agent might do particularly well against those agents. Another explanation could be that the remaining teams modified their agents to bid lower in the final round or made other improvements after seeing how the competition reacted to their agent. Unlike some of the other teams, we did not take advantage of the opportunity to alter our agent between the semifinals and the finals.

V. RELATED WORK

Although genetic algorithms have been used before in other types of auction mechanisms (see e.g. [11], [13], [14]), to the best of our knowledge, we are the first to propose the use of evolutionary computation techniques to keyword bidding for sponsored search. The rapid growth in the online advertising industry and its relationship with search engines is attracting a growing research interest in the topic of online keyword based ad auctions, particularly towards per-click pricing rather than per-impression strategies [5]. Ongoing research efforts include the study of sponsored search auction mechanisms with ranking of advertisements as a whole (see e.g. [1], [2]) as well as the proposal of different strategies to optimize the advertiser's profit. Kitts and Leblanc [9] for instance treat the problem of finding CPC offers that will optimize the profit as a linear programming problem with unknown parameters that need to be estimated, such as the expected number of clicks generated by users overall, and the slot position that can be expected for a specific CPC offer. Cary et. al. [3] study the revenue, convergence and robustness of greedy bidding strategies, while Liang and Qi [10] investigate cooperative and vindictive strategies.

Not much is publicly known yet about the agents participating in the 2009 TAC/AA competition, because most participants have not yet disclosed their approaches. In fact, in this current paper we disclose our approach for the first time. The TacTex agent from the University of Texas used particle filters to predict the bid necessary to achieve a certain position, and the number of clicks and conversions for a given position. Shlemazl, a Brown University agent, was based largely on mathematical modeling using rule based techniques. Aston University's agent was based on case-driven logic to handle individual conditions in the auction. It

[6]Available for download at http://www.sics.se/tac/showagents.php?id=132

used a combination of strategies depending on current profit and average position for each query.

VI. Conclusion and Future Work

Web search engines have advertising platforms that allow advertisers to bid for visible ad space based on the keywords entered as queries by search engine users. From an advertiser's perspective, devising an optimal bidding strategy to maximize profits by keeping costs per click down and achieving high conversion rates from the potential customers is an extremely challenging task. Thus automated bidding agent frameworks have a tremendous role to play in the coming years to help advertisers manage their ad campaigns to generate revenue. Such automation frameworks are therefore key to the next generation of electronic commerce.

In this paper we have presented the first ever genetic algorithm based solution to learn to optimize bid bundles in sponsored search auctions. Even though the results are presented in the context of the TAC Ad Auctions competition hosted at IJCAI 2009, the solution is fairly generic since the competition itself strongly mimics real-world search engine keyword based ad auction scenarios. The main merit of our *DNAgent* (3rd in the semifinals, see Table III) and (5th in the finals, see Table IV) is its simplicity in terms of design effort compared to other approaches requiring a deep knowledge of the application field.

Given the novelty of the domain for evolutionary computing, there are a few notable areas for future work that can be explored. Performing feature selection to determine key input features that result in optimal slot positions is an interesting area of exploration. Deciding on the parameters to encode in the chromosome is particularly difficult considering the multitude of information available in the daily reports given to the agent. Ideally, one would want to include all available information since this would best equip the agent to choose the best action. Unfortunately, the more information included, the larger the chromosome length, leading to greater learning complexity. We chose to use slot position because it is easily encoded into a small number of discrete values and it also strongly correlates with other features such as number of impressions, number of clicks, and cost per click. Specialization matching is easily discretized, and offers a way of categorizing queries in terms of their revenue generation capabilities. Finally, taking into account the backorder quantity is helpful to keep an advertiser from spending money on top ranked slot positions when he does not have enough products in stock to fulfill potential orders anyway.

One particular kind of features that we have not yet explored are those that capture competitor behavior. In the real world, advertiser strategies are a function of both private and public information. Though the search engine weights each advertiser for its preference to show the ad in a given slot, the advertisers themselves can also gather information about competitor behavior and position history, such as the type of ads displayed by each advertiser for a query and the average positions of all of advertisers given by the ad auction for a query. All this information can be readily utilized to optimize CPC offers. In this paper, we did not use any explicit information about other advertisers.

Similarly as in [14], we discovered that co-evolving a population by running internal competitions with no external agents involved can have some negative side effects. Occasionally multiple members of the population appeared to develop cooperative strategies, such as all bidding low. This kept the overall market price for slot positions low, allowing each agent to achieve a higher profit. This is an interesting phenomenon, but it is outside the scope of TAC/AA competition, because agents are not allowed to intentionally cooperate. In the future, we intend to train our agents through competition against external agents, such as the strongest agents from the 2009 TAC/AA competition. We expect that a definition of the fitness function based on average profit obtained while competing against other strategies will give rise to even stronger genetically evolved agents.

References

[1] G. Aggarwal, A. Goel, and R. Motwani, "Truthful Auctions for Pricing Search Keywords", *Proceedings of the 7th ACM Conference on Electronic Commerce (EC'06)*, pp. 1-7, 2006.

[2] C. Borgs, J. Chayes, O. Etesami, N. Immorlica, K. Jain, and M. Mahdian. "Dynamics of Bid Optimization in Online Advertisement Auctions", *Proceedings of the 16th International World Wide Web Conference (WWW2007)*, pp. 531-540, 2007.

[3] M. Cary, A. Das , B. Edelman, I. Giotis, K. Heimerl, A. R. Karlin, C. Mathieu, and M. Schwarz. "Greedy Bidding Strategies for Keyword Auctions", *Proceedings of the 8th ACM Conference on Electronic Commerce (EC'08)*, pp. 262-271, 2007.

[4] B. Edelman, M. Ostrovsky, and M. Schwarz. "Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords", *American Economic Review*, vol. 97, pp. 242-259, 2007.

[5] D.L. Hoffman and T.P. Novak, "How to Acquire Customers on the Web", *Harvard Businesses Review*, vol. 78(3), pp. 179-188, 2000.

[6] F. Herrera, M. Lozano, and J. L. Verdegay. "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis", *Artificial Intelligence Review*, vol. 12, pp. 265-319, 1998.

[7] P. R. Jordan, B. Cassell, L. F. Callender, and M. P. Wellman, "The Ad Auctions Game for the 2009 Trading Agent Competition", Specification Version 0.9.6, 2009, http://aa.tradingagents.org/documentation/, accessed on Jan 15, 2009.

[8] P. R. Jordan and M. P. Wellman, "Designing an Ad Auctions Game for the Trading Agent Competition", *IJCAI-09 Workshop on Trading Agent Design and Analysis (TADA-09)*, 2009.

[9] B. Kitts and B. Leblanc. "Optimal Bidding on Keyword Auctions", *Electronic Markets*, vol. 14(3), 2004.

[10] L. Liang and Q. Qi. "Cooperative or Vindictive: Bidding Strategies in Sponsored Search Auction", *Lecture Notes in Computer Science*, vol. 4858, pp. 167-178, 2007.

[11] S. Phelps, K. Cai, P. McBurney, J. Niu, S. Parsons, and E. Sklar. "Auctions, Evolution, and Multi-agent Learning", *Lecture Notes in Computer Science*, vol. 4865, pp. 188-210, 2008.

[12] P. Rusmevichientong and D. P. Williamson, "An Adaptive Algorithm for Selecting Profitable Keywords for Search-based Advertising Services", *Proceedings of the 7th ACM conference on Electronic Commerce (EC'06)*, pp.260–269, 2006.

[13] Y. Saez, D. Quintana and P. Isasi. "Effects of a Rationing Rule on the Ausubel Auction: A Genetic Algorithm Implementation", *Computational Intelligence*, vol.23(2), pp. 221-235, 2007.

[14] I. Walter and F. Gomide. "Multiagent Coevolutionary Genetic Fuzzy System to Develop Bidding Strategies in Electricity Markets: Computational Economics to Assess Mechanism Design", *Evolutionary Intelligence*, vol. 2, pp. 53-71, 2009.